Package 'GHAgricProductivityLab'

November 11, 2025

Type Package

Title Agricultural Productivity in Ghana

Version 0.0.0.9000

Author Francis Tsiboe [aut, cre] (<https://orcid.org/0000-0001-5984-1072>)

Maintainer Francis Tsiboe <ftsiboe@hotmail.com>

Contributor -

Reviewer -

Creator Francis Tsiboe

Description Provides tools and datasets for investigating the drivers of agricultural production shortfalls in Ghana.
It compiles research studies that examine farmer-specific and institutional factors to assess whether inefficiencies arise from technical inefficiency, technology gaps, or both. The package offers empirical evidence to inform policy discussions and interventions aimed at improving agricultural productivity, particularly in contexts with limited access to modern technologies.

License GPL-3 + file LICENSE

URL https://github.com/ftsiboe/GHAgricProductivityLab

BugReports https://github.com/ftsiboe/GHAgricProductivityLab/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

Depends R (>= 4.1.0)

Imports MatchIt, data.table, haven

Remotes github::dylan-turner25/rfcip

Suggests dplyr, knitr, crayon, tidyr, withr, rmarkdown, cobalt, dplyr,doBy, stringr, testthat (>= 3.0.0)

LazyData true

Cite-us If you find it useful, please consider staring the repository and citing the following studies
- Tsiboe, F. and Turner, D. (2025). ``Incorporating buy-up price loss coverage into the United States farm safety net." Applied Economic Perspectives and Policy.
- Tsiboe, F., et al. (2025). ``Risk reduction impacts of crop insurance in the United States."

Applied Economic Perspectives and Policy.
- Gaku, S. and Tsiboe, F. (2024). Evaluation of alternative farm safety net program combination strategies. Agricultural Finance Review.

Contents

`clear_rfcipCalcPass_cache`

Clear the package cache of downloaded data files

Description

Deletes the entire cache directory used by the **rfcipCalcPass** package to store downloaded data files. Useful if you need to force re-download of data, or free up disk space.

Usage

```
clear_rfcipCalcPass_cache()
```

Value

Invisibly returns `NULL`. A message is printed indicating which directory was cleared.

See Also

Other helpers: `rfcipCalcPass_control()`

Examples

```
## Not run:
# Remove all cached data files so they will be re-downloaded on next use
clear_rfcipCalcPass_cache()

## End(Not run)
```

covariate_balance *Compute Covariate Balance Summaries Across Matching Specs*

Description

Compute Covariate Balance Summaries Across Matching Specs

Usage

```
covariate_balance(match_specifications, matching_output_directory)
```

Arguments

match_specifications	data.frame/data.table with at least columns: boot, ARRAY, and the spec fields stored in RDS (e.g., method, distance, link).
matching_output_directory	Directory containing RDS files named as "0001.rds", "0002.rds", etc.

Details

Reads each matching result RDS (expected to contain m.out and match_specifications), extracts balance via cobalt::bal.tab, reshapes, computes a composite balance $rate = mean((Diff - 0)^2, (KS - 0)^2, (V_Ratio - 1)^2)$, and averages by ARRAY, method, distance, link, sample.

Value

A list with:

rate	data.frame of mean balance metrics by spec (Adj sample only) with a composite rate.
bal_tab	long-format balance table per covariate/stat/sample/spec.

draw_matched_samples *Draw matched samples (stratified bootstrap + matching)*

Description

Performs stratified resampling at the Surveyx, EaId level (excluding EaIds listed for the current bootstrap ID) and computes adjusted sampling weights used for matching. Then fits a matching model per match_specifications[i,] using **MatchIt**, with exact matching on Emch and distance/link from match_specifications.

Usage

```
draw_matched_samples(
  i,
  data,
  match_variables_exact,
  match_variables_scaler,
  match_variables_factor,
  match_specifications,
  sample_draw_list,
  verbose = FALSE
)
```

Arguments

i	Integer index selecting the row of match_specifications to use.
data	A data.frame/data.table containing at least: Surveyx, EaId, HhId, Mid, UID, Weight, Treat, plus columns named in Emch, Scle, Fixd.
match_variables_exact	A character vector of variable names to be included in the exact match component (e.g., "gender", "region").
match_variables_scaler	A character vector of continuous or scalar variable names to include as covariates in the general match formula (e.g., "age", "income").
match_variables_factor	A character vector of factor variable names to be included as dummy-coded categorical covariates in the general match formula.
match_specifications	Data frame of matching specifications with columns boot, method, distance, and optionally link.
sample_draw_list	Data frame where column ID identifies the bootstrap draw, and each remaining column corresponds to a Survey containing the sampled EaId.
verbose	Logical; if TRUE, prints the chosen matching method and timing. Default FALSE.

Details

Adjusted weights are computed as $pWeight = Weight \times (alloc/allocj)$, where alloc is the pre-exclusion sum of Weight by Surveyx, EaId and allocj is the post-exclusion sum.

Exact matching is performed on variables in Emch. The distance model formula is constructed as Treat ~ Scle + Fixd. When distance == "glm", m.order = "largest" is used; otherwise "closest".

Value

A list with:

match_specifications The selected row from match_specifications.

m.out The matchit object.

md Matched data: Surveyx, EaId, HhId, Mid, UID, weights, pWeight.

df The analysis data with adjusted weights: Surveyx, EaId, HhId, Mid, UID, pWeight.

`draw_match_sample_specifications`
Draw / Match Sample Specifications

Description

Generates (i) a draw list by sampling `EaId` within each unique `Survey` group and (ii) a grid of matching specifications for each bootstrap draw.

Usage

```
draw_match_sample_specifications(drawN, data, myseed = 3242025)
```

Arguments

`drawN`	Integer. The number of draws to perform per `Survey`.
`data`	A data.frame or data.table containing at least the columns `Survey` and `EaId`.
`myseed`	Integer. Seed for random number generation (default `03242025`).

Details

Draw list: For each unique value of `Survey`, the function samples `drawN` `EaId` values with replacement and prepends a 0 row (ID = 0) for the baseline. The result is then spread to wide format with one column per `Survey`.

Matching specs: For each draw `ID`, creates a set of matching model specifications that include:

- Nearest neighbor with distances: `"euclidean"`, `"scaled_euclidean"`, `"mahalanobis"`, `"robust_mahalanobis"`.
- Nearest neighbor with distance `"glm"` and links: `"logit"`, `"probit"`, `"cloglog"`, `"cauchit"`.

An `ARRAY` index is added for convenience.

Value

A list with two elements:

`m.specs` A data.frame of matching specifications with columns `boot`, `method`, `distance`, `link`, `ARRAY`.

`drawlist` A data.frame in wide format where each `Survey` is a column and rows correspond to draw `ID` (0:`drawN`).

Note

Requires that `data` contain `Survey` and `EaId`. `tidyr::spread()` is used for wide reshaping (consider `tidyr::pivot_wider()` in new code).

`functional_forms` *Generate Functional and Distribution Forms*

Description

Creates symbolic production function strings for stochastic frontier analysis (SFA), including Cobb-Douglas (CD) and Translog (TL) specifications, plus a set of candidate distributions for the inefficiency term.

Usage

```
functional_forms(nX = 5, trend = FALSE)
```

Arguments

`nX`	Integer. Number of input variables. Default is 5.
`trend`	Logical. If `TRUE`, adjusts the (optional) transcendental form (`TP`) by removing the last linear input term. Default is `FALSE`.

Details

The returned `fxnforms` list includes:

- `CD`: sum of log inputs (`lnI1 + lnI2 + ...`).
- `TL`: CD terms plus second-order and pairwise interaction terms (e.g., `0.5 * lnIi^2` and `lnIi * lnIj` for `i < j`).

Additional forms (Linear, Quadratic, Generalized, Transcendental) are shown in the code as commented examples; uncomment to include them. The `trend` argument only affects `TP` when that form is enabled.

`distforms` lists common inefficiency distributions (e.g., half-normal, truncated normal, exponential, lognormal, Weibull) and whether a scaling property is assumed for each.

Value

A list with:

`fxnforms` Named character strings of functional forms.

`distforms` Named list of inefficiency distributions with a `scaling` flag.

get_crop_area_list *Extract Matching Crop Area Columns from a Dataset*

Description

Identifies and returns the column names in a dataset corresponding to crop area variables for a specified set of crops. The function looks for column names that start with "Area_" and match any of the crops provided in selected_crops.

Usage

```
get_crop_area_list(
  data,
 selected_crops = c("Beans", "Cassava", "Cocoa", "Cocoyam", "Maize", "Millet", "Okra",
    "Palm", "Peanut", "Pepper", "Plantain", "Rice", "Sorghum", "Tomatoe", "Yam")
)
```

Arguments

data	A data.frame or data.table containing crop-related variables. Column names are expected to include fields prefixed with "Area_", such as "Area_Maize".
selected_crops	A character vector specifying the crop names to filter. Defaults to a common set of crops including "Beans", "Cassava", "Cocoa", "Cocoyam", "Maize", "Millet", "Okra", "Palm", "Peanut", "Pepper", "Plantain", "Rice", "Sorghum", "Tomatoe", and "Yam".

Value

A character vector containing the names of columns in data that correspond to the specified crop area variables (e.g., "Area_Maize", "Area_Rice"). Returns an empty vector if no matching columns are found.

harmonized_data_prep *Prepare Data for Agricultural Productivity Analysis*

Description

Cleans and transforms a dataset by creating new variables, applying log transformations, converting selected variables to factors or characters, and recoding education levels. The function is designed to standardize inputs for further analysis of agricultural productivity.

Usage

```
harmonized_data_prep(data)
```

Arguments

data	A data.frame or data.table containing household- and farm-level variables such as weights, demographic information, and agricultural inputs.

Value

A cleaned and transformed `data.frame` or `data.table` with additional variables ready for analysis.

`rfcipCalcPass_control` *Create a control list of adjustment factors for PASS Calculators*

Description

This function initializes a named list of control parameters (adjustment factors) used throughout the farm policy simulation pipeline. Each element has a sensible default but can be overridden to customize behavior.

Usage

```
rfcipCalcPass_control(
  revenue_lookup_adjustment_factor = 1,
  unit_structure_discount_factor = 1,
  additive_optional_rate_adjustment_factor = 0,
  multiplicative_optional_rate_adjustment_factor = 1,
  capped_revenue_add_on_factor = 0,
  liability_adjustment_factor = 1,
  multiple_commodity_adjustment_factor = 1,
  reported_acres = 1,
  insured_share_percent = 1,
  price_election_percent = 1,
  damage_area_rate = 1,
  harvest_price_inclusion_plans = c(2, 5, 16, 32, 88),
  non_price_risk_plans = c(1, 90, 4, 31, 87),
  rma_rounding = TRUE,
  yield_ratio_cup_and_cap = TRUE,
  continuous_integration_session = FALSE,
  adm_decoy_state_abb = "ND"
)
```

Arguments

`revenue_lookup_adjustment_factor`
: Numeric scalar. Multiplier applied to revenue look ups. (Default = 1)

`unit_structure_discount_factor`
: Numeric scalar. Discount factor for unit structure. (Default = 1)

`additive_optional_rate_adjustment_factor`
: Numeric scalar. Additive adjustment to optional rates. (Default = 0)

`multiplicative_optional_rate_adjustment_factor`
: Numeric scalar. Multiplicative adjustment to optional rates. (Default = 1)

`capped_revenue_add_on_factor`
: Numeric scalar. Add-on factor applied to capped revenue. (Default = 0)

`liability_adjustment_factor`
: Numeric scalar. Multiplier applied to liability coverage. (Default = 1)

`multiple_commodity_adjustment_factor`
: Numeric scalar. Adjustment factor when multiple commodities are insured. (Default = 1)

`reported_acres`
: Numeric scalar. Number of acres reported for insurance purposes. (Default = 1)

`insured_share_percent`
: Numeric scalar. Share of the crop insured (0-1). (Default = 1)

`price_election_percent`
: Numeric scalar. Proportion of the elected price used (0-1). (Default = 1)

`damage_area_rate`
: Numeric scalar. Rate applied to damage-area calculation.(Default= 1)

`harvest_price_inclusion_plans`
: Vector of plan codes that include harvest price in guarantee calculation.

`non_price_risk_plans`
: Vector of plan codes that do not adjust revenue based on harvest price.

`rma_rounding`
: logical(1) or numeric(1) If FALSE, rounds only to integer. Otherwise multiplies the number of digits by this factor (mimics round(x, n * rma_rounding)). Defaults to TRUE.

`yield_ratio_cup_and_cap`
: logical(1) If TRUE, enforces a 0.50-1.50 `cup` & `cap` on yield ratios. Defaults to TRUE.

`continuous_integration_session`
: logical(1). If TRUE, a small deterministic subset of the Actuarial Data Master (ADM) YTD ZIP archive is used. This is designed to be safe and fast for use in continuous integration sessions. see `build_min_adm()`

`adm_decoy_state_abb`
: tate abbreviation indicating which state's decoy ADM to use (default is ND).

Value

A named list of all control parameters, ready to be passed to other simulation functions.

See Also

Other helpers: `clear_rfcipCalcPass_cache()`

Examples

```
## Not run:
# Use all defaults:
ctrl <- rfcipCalcPass_control()

# Override a couple of factors:
ctrl2 <- rfcipCalcPass_control(
  free_acres_factor = 0.15,
  liability_adjustment_factor = 0.9
)


## End(Not run)
```

study_setup *Initialize Study Environment and Directory Structure*

Description

Initialize Study Environment and Directory Structure

Usage

```
study_setup(
  seed = 1980632,
  project_name,
 local_directories = list(output = file.path("replications", project_name, "output"),
    matching = file.path("replications", project_name, "output", "matching"),
    treatment_effects = file.path("replications", project_name, "output",
  "treatment_effects"), estimations = file.path("replications", project_name, "output",
  "estimations"), figure_data = file.path("replications", project_name, "output",
  "figure_data"), figure = file.path("replications", project_name, "output", "figure"))
)
```

Arguments

seed	Numeric/integer scalar seed. Will be coerced to integer. Default: 1980632.
project_name	Length-1, non-NA character project name (required).
local_directories	Named list of character paths to create. Defaults use project_name.

Value

List with wd (directories) and seed.

write_match_formulas *Construct Matching Formulas for Exact and General Matching*

Description

Builds two types of matching formulas commonly used in propensity score or covariate matching workflows:

- An **exact match** formula specifying categorical variables to match exactly.
- A **general match** formula specifying numeric and factor covariates for distance-based matching.

The function returns both formulas as character strings that can be used in matching functions such as MatchIt::matchit() or Matching::Match().

Usage

```
write_match_formulas(
  match_variables_exact,
  match_variables_scaler,
  match_variables_factor
)
```

Arguments

`match_variables_exact`
: A character vector of variable names to be included in the exact match component (e.g., `"gender"`, `"region"`).

`match_variables_scaler`
: A character vector of continuous or scalar variable names to include as covariates in the general match formula (e.g., `"age"`, `"income"`).

`match_variables_factor`
: A character vector of factor variable names to be included as dummy-coded categorical covariates in the general match formula.

Value

A named list with two elements:

`exact_match` A string representing the exact match formula (factor variables only).

`general_match` A string representing the general match formula, including both continuous and factor covariates on the right-hand side of `Treat ~`.

Index